Desenvolvendo Use Cases a partir de Modelagem Organizacional

Victor F.A. Santander[1] **Jaelson F. B. Castro**[2]

Universidade Federal de Pernambuco – Centro de Informática
Cx. Postal 7851, CEP 50732-970, Recife-PE, BRAZIL
Phone: (+55 81) 271-8430, Fax: (+55 81) 271-8438
{vfas,jbc}@cin.ufpe.br

Resumo: O paradigma de desenvolvimento orientado a objetos tem conquistado fervorosos adeptos na comunidade de engenharia de software. Um dos mais importantes avanços foi o surgimento da UML, uma linguagem padronizada para modelagem visual. Tipicamente Diagramas de USE CASES tem sido usados para capturar **requisitos funcionais** do sistema a ser desenvolvido. Contudo, o desenvolvimento de sistemas computacionais ocorre dentro de um contexto onde processos organizacionais estão bem estabelecidos. Portanto, é preciso capturar os **requisitos organizacionais** para definir como o sistema pretendido irá satisfazer os objetivos da organização, por que ele é necessário, quais as alternativas existentes, quais as implicações das alternativas para as várias partes interessadas, etc. Lamentavelmente UML e técnicas baseadas em cenários em geral, não estão equipadas para modelar os requisitos organizacionais. Precisamos de outra técnica, tal como **i***, para representar estes aspectos. Contudo, os requisitos organizacionais precisam ser relacionados aos requisitos funcionais, representados através de Diagramas de Use Case. Neste artigo apresentaremos algumas diretrizes que podem auxiliar o engenheiro de requisitos a desenvolver Diagramas de Use Case em UML a partir dos modelos organizacionais propostos na técnica i*.
Palavras-Chave: Cenários, Modelagem Organizacional, Engenharia de Requisitos

Abstract: The object oriented development paradigm has attracted many supporters in the Software Engineering community. One of the most important advance was the Unified Language Modelling (UML), a standard for visual modelling. Use Cases Diagrams have been used for capturing system functional requirements. However, system development occurs in a context where organisation processes are well established. Therefore, we need to capture organisational requirements to define how the system fulfils the organisation goals, why it is necessary, what are the possible alternatives, what are the implications to the involved parts, etc. Unfortunately, UML and other scenario-based techniques not are well equipped for modelling organisational requirements. We need others techniques, such as i*, to represent this aspects. Nevertheless, organisational requirements must related to functional requirements represented as Use Cases. In this paper we present some guidelines to assist the requirement engineer in the development of use cases from the organizational models represented by i* technique.
Keywords: Scenarios, Organisational Modelling, Requirement Engineering.

---

[1] Professor da UNIOESTE. Aluno de Doutorado na UFPE/CIN. Parcialmente apoiado pelo CNPq Proc. No. 147192/1999-4.

[2] Este trabalho foi realizado durante a visita do autor ao Department of Computer Science, University of Toronto. O trabalho foi parcialmente apoiado pelo CNPq Proc. 203262/86-7 (NV).

1. Introdução

O desenvolvimento de softwares cada vez mais complexos, passíveis de certificação e com menor custo possível, tem-se tornado um desafio constante para a comunidade de engenharia de software. Diversas técnicas, metodologias e ferramentas vêm sendo propostas com o intuito de suportar e auxiliar a produção de software de qualidade. Neste contexto, uma das etapas mais críticas está relacionada à engenharia de requisitos. Freqüentemente, requisitos de software são inadequadamente elicitados, analisados e especificados, sendo estes fatores decisivos para o desenvolvimento de softwares de baixa qualidade. Solucionar problemas de incompletude e inconsistência de requisitos ainda nas etapas iniciais do processo de desenvolvimento, possibilita a obtenção de produtos de softwares mais confiáveis e diminui os custos futuros de manutenção.

O desenvolvimento de novas técnicas de suporte às atividades da engenharia de requisitos tem sido uma preocupação atual da comunidade acadêmica e industrial. Diversas abordagens para elicitar, analisar e especificar requisitos tem surgido desde as primeiras técnicas utilizada com esse fim. No início, a análise estruturada de sistemas através de técnicas tais como DFDs (Diagramas de fluxo de Dados), DD (Dicionários de Dados) e DER (Diagramas de Entidade e relacionamentos) procurava atentar para a análise de requisitos. Atualmente, técnicas tais como cenários, use cases, etnografia, entrevistas, prototipação e outras, também são apontadas como alternativas para suportar as atividades de engenharia de requisitos.

Dentre estas abordagens, técnicas baseadas em cenários tem recebido uma atenção especial. Uma das abordagens de cenários que tem se destacado é Use Cases. Esta técnica é parte integrante e chave na Linguagem de Modelagem Unificada (UML) (Booch et al. 1999), a qual é um padrão de linguagem de modelagem para o desenvolvimento de software orientado a objetos. Outras propostas existentes de desenvolvimento de cenários são apresentadas no projeto CREWS (Ralyté 1999) (Ralyté et al. 1999) em Leite et al. (1997).

Apesar do consenso e do reconhecimento de cenários como ferramenta importante no processo de engenharia de requisitos, podemos apontar algumas carências da técnica, principalmente no que diz respeito a inclusão de aspectos inerentes ao ambiente organizacional no qual o software está inserido. Quando usuários desejam desenvolver um software, em grande parte dos casos não há uma idéia concreta do que os mesmos desejam. Geralmente, o que se tem, são intenções e desejos de facilitar a execução de atividades no ambiente organizacional. Por outro lado, o processo de engenharia de requisitos atenta inicialmente para a necessidade de elicitar os requisitos do sistema junto ao usuário, procurando obter o que o mesmo deseja e espera do sistema. Muitos dos problemas associados com o desenvolvimento de software podem iniciar neste fase, pois detectar o que é realmente relevante para o usuário levando-se em consideração os objetivos organizacionais, não é uma tarefa trivial. Técnicas baseadas em cenários auxiliam nesta tarefa mas precisam ser complementadas.

É bastante aceito que o trabalho da engenharia de requisitos pode ser melhorado se modelarmos aspectos organizacionais visando entender melhor as intenções e motivações organizacionais que incorporam o desejo do desenvolvimento de um software. Neste sentido, várias propostas objetivando a modelagem organizacional tem sido apontadas (Yu 1995) (Bubenko 1993). Este tipo de modelagem objetiva

fornecer recursos que permitam modelar as intenções, relacionamentos e motivações entre os membros de uma organização. A partir destes modelos permite-se compreender melhor o funcionamento do ambiente organizacional bem como as relações humanas e de trabalho entre os participantes da organização. Com estas informações, os requisitos de uma solução computacional para processos organizacionais podem ser melhor elicitados e especificados.

Tendo em vista este panorama, objetivamos estudar de que forma o desenvolvimento de um modelo organizacional através da técnica i* (Yu 1995), pode complementar e servir de fonte de informação para o desenvolvimento de cenários. O estudo consiste basicamente em estabelecer um correlação entre os elementos em **i*** e os elementos componentes de cenários sob a forma de **Use Cases** (Schneider et al. 1998). Motiva-nos o fato de podermos evoluir de uma modelagem organizacional envolvendo o domínio de um sistema computacional a ser desenvolvido, para uma elicitação de requisitos através de Use Cases bem como para uma especificação de requisitos mais completa e não ambígua. A idéia é subsidiar engenheiros de requisitos com metas estratégicas de negócios na organização, as quais devem ser analisadas e ponderadas no momento do desenvolvimento de um sistema computacional. Modelos organizacionais permitem analisar a organização, seu processos, as influências positivas e negativas bem como possíveis impactos para a organização decorrentes do desenvolvimento de sistemas computacionais. Com base nesta análise, o esforços podem ser direcionados na captura da funcionalidade destes sistemas adotando técnicas tais como use cases/cenários.

Assim, a partir da correlação entre as técnicas i* e Use Cases, fatores do ambiente organizacional inicialmente não considerados em Use Cases, podem ser integrados aos mesmos.

Além desta introdução, dividimos o artigo da seguinte forma: na seção 2 apresentamos uma descrição da técnica de modelagem organizacional **i***. Na seção 3 descrevemos os principais conceitos associados com técnicas baseadas em cenários com ênfase na descrição de **Use Cases**. Na seção 4 apresentamos algumas diretrizes que visam auxiliar o desenvolvimento de Use Cases a partir de modelos organizacionais e aplicamos estas diretrizes a um estudo de caso. Na seção 5 apresentamos as consideração finais do trabalho bem como os trabalhos futuros.

2. A técnica i*

A técnica i* objetiva possibilitar a representação/modelagem de aspectos organizacionais envolvidos com processos (Yu 1995). Basicamente permite descrever aspectos de intencionalidade e motivações envolvendo atores em um ambiente organizacional. Para descrever estes aspectos são propostos dois modelos: O Modelo de Dependências Estratégicas (**SD**) e o Modelo de Razões Estratégicas (**SR**).

2.1. Modelo de Dependências Estratégicas (SD).

O Modelo de Dependências Estratégicas é composto por **nós** e **ligações**. O nós representam os **atores** no ambiente e as ligações são as **dependências** entre os atores. Por **ator** entende-se uma entidade que realiza ações para obter objetivos no contexto do ambiente organizacional. Atores dependem uns dos outros para obter objetivos. O ator que depende de alguma forma de outro ator é chamado de **Depender** e o ator que

atende e satisfaz o Depender é denominado de **Dependee.** O objeto ou elemento de dependência entre Depender e Dependee é denominado de **Dependum**.

As dependências apresentadas neste modelo podem ser de diferentes tipos, tendo como base o tipo do Dependum. A seguir descrevemos os quatro tipos de dependências propostos em i*:

- Dependência de **Objetivo**: O Depender depende do Dependee para modificar um estado do mundo. O Dependee tem como papel obter um objetivo para o Depender. O Dependee pode obter este objetivo e modificar e estado atual escolhendo a forma como isto será feito, tendo liberdade para tomar as decisões necessárias para a obtenção do objetivo. No entanto, o Depender torna-se vulnerável pois o Dependee pode falhar em atingir este objetivo.
- Dependência de **Tarefa**: O Depender depende do Dependee para executar uma atividade, sendo de responsabilidade do Depender mostrar o caminho como isto será feito. No entanto, não é fornecido para o Dependee "o porque" da realização da tarefa. O Depender torna-se vulnerável pois o Dependee pode falhar na realização da tarefa pois o Dependee pode não seguir os passos indicados para realizar a tarefa ou pode estabelecer outras prioridades antes de realizá-la. Neste tipo de dependência, o Depender toma as decisões e os objetivos do Depender não são de conhecimento do Dependee.
- Dependência de **Recurso**: O Depender depende do Dependee em relação à disponibilidade de uma entidade (recurso) seja ela física ou de informação. Isto significa que o Dependee deve fornecer um recurso que o Depender necessita para realizar outras atividades no ambiente organizacional. O Depender torna-se vulnerável pois o recurso pode não ser disponibilizado pelo Dependee. Neste tipo de dependência os aspectos relacionados com decisões não são considerados pois o que está em questão é a disponibilidade de um recurso ou não.
- Dependência de **Objetivo-soft**: Define-se objetivo-soft como um objetivo cuja avaliação de realização é bastante subjetiva e o seu significado não é claramente conhecido. Assim não se pode afirmar objetivamente se o mesmo foi satisfeito ou não. Geralmente, o entendimento e avaliação do objetivo-soft acontece ao longo do processo da realização de tarefas associadas com o objetivo-soft. Estes objetivos-soft são também referenciados como requisitos não funcionais no contexto de engenharia de requisitos. O Depender depende do Dependee para realizar alguma tarefa que satisfaça o objetivo-soft. A decisão se o objetivo é ou não satisfeito é tomada pelo Depender com base nas atividades realizadas pelo Dependee. O Depender torna-se vulnerável pois o Dependee pode falhar na obtenção das condições que satisfazem o objetivo-soft.

Para entender melhor os conceitos associados com o modelo SD, vejamos o exemplo apresentado na figura 1 retirado de (Yu 1995). Este exemplo modela as dependências estratégicas em um processo organizacional que pode ser adotado para realizar agendamentos de reuniões. O ator Meeting Initiator (Agendador) possui uma série de dependências em relação ao ator Meeting Participant (Participante). As dependências do tipo recurso *ExclusionDates(p) e PreferredDates(p*) identificam que

o participante deve fornecer informações que incluem um conjunto de datas que o mesmo exclui na possibilidade de agendamento da reunião e um conjunto de datas que são preferidas pelo mesmo para agendamento, respectivamente. O ator Meeting Initiatior (Agendador) depende destes recursos para continuar o processo de agendamento.

Por outro lado, Meeting Participant (Participante) depende de uma data de agendamento proposta *(ProposedDate(m))* pelo Meeting Initiator (Agendador) para poder concordar ou não com a data. A dependência do tipo objetivo *AttendsMeeting(p,m)* considera que Meeting Participant deve satisfazer o objetivo de atender à reunião. De acordo com o conceito associado com este tipo dependência, o ator Meeting Participant pode escolher a forma como este objetivo será obtido. A dependência *Agreement(m,p)* aponta para um recurso que Meeting Participant deve fornecer, representando uma resposta do participante concordando ou não com a data proposta pelo Meeting Initiator para a reunião. A dependência do tipo objetivo *AttendMeeting(ip,m)* entre Meeting Initiator (Agendador) e Important Participant (Participante Importante) indica uma dependência crítica (representada pelo símbolo "X" do lado de Meeting Initiator) do ator Meeting Initiator em relação a necessidade de que Important Participant atenda ao encontro.

A dependência do tipo objetivo-soft *Assured(AttendsMeeting(ip,m))* é uma referência a um objetivo-soft que tem uma avaliação subjetiva por parte do Depender Meeting Initiator, indicando, por exemplo, se a garantia que um participante importante participar ao encontro é **satisfatória**.

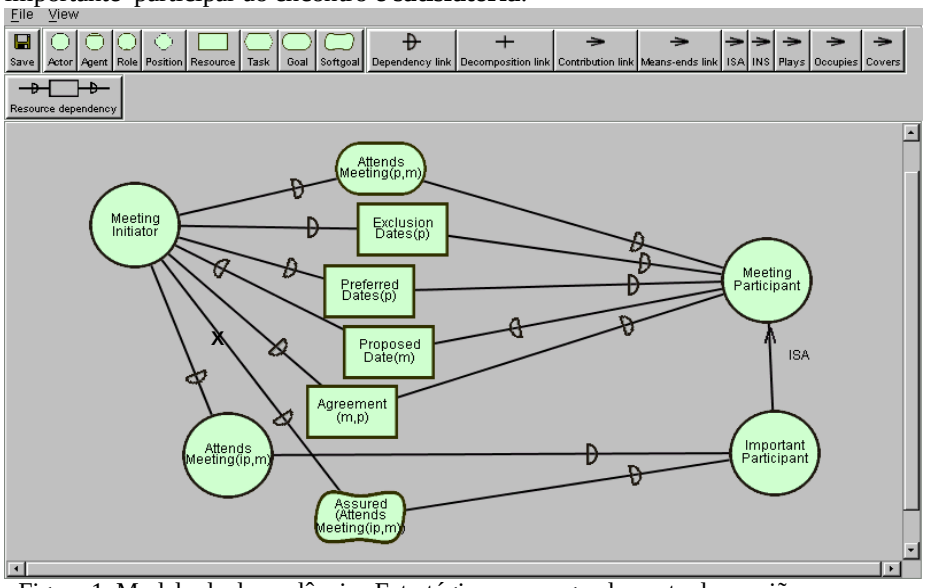

Figura 1. Modelo de dependências Estratégicas para agendamento de reuniões.

As dependências no modelo SD podem existir em maior ou menor grau de rigorosidade. Como já observado no exemplo acima, a dependência AttendMeeting(ip,m) entre Meeting Initiator e Important Participant indica uma dependência do tipo crítica (sinalizada pelo símbolo "X"). Como neste caso, uma forte

dependência por parte do Depender indica um alto grau de vulnerabilidade do Depender em relação a um Dependum. Se a forte dependência está relacionada com o Dependee, indica que o mesmo deve esforçar-se ao máximo para satisfazer o Dependum relacionado com o Depender.

Para aumentar o grau de compreensão das dependências entre os atores no ambiente organizacional, o modelo de dependências estratégicas permite a subdivisão dos atores em subunidades relacionadas. Permite-se que um ator possa ser subdividido em elementos tais como: agentes, papéis e posições. No entanto, por questões de limites de espaço, neste artigo consideraremos o ator na sua forma mais completa, considerado-o um ator social.

Tendo como base estes elementos definidos para o modelo de dependências estratégicas, tanto as intenções quanto motivações e objetivos organizacionais podem ser modelados e várias alternativas para o desenvolvimento de sistemas computacionais podem ser avaliadas optando-se por aquela que melhor satisfaça os objetivos de todos os stakeholders[3].

A seguir descrevemos resumidamente o outro modelo que compõe a técnica i*: o modelo de Razões Estratégicas (SR).

2.2. O modelo de Razões Estratégicas (SR).

O modelo de Razões Estratégicas é um modelo complementar ao modelo de dependências estratégicas apresentado na seção anterior. Este modelo permite compreender e modelar de forma mais detalhada as razões associadas com cada ator e suas dependências. Estas razões estão associadas com a forma que procedimentos e ações internas inerentes a elementos de processos podem ser representadas. Cada ator externamente possui dependências com outros atores mas internamente possui objetivos e rotinas que impulsionam e justificam cada dependência.

O princípio básico para desenvolver este modelo é observar as razões que estão associadas com cada ator em relação aos relacionamentos de dependência com outros atores. Um bom caminho para iniciar a decomposição é observar como os Dependee podem satisfazer os dependums associados com os mesmos e a partir desse ponto observar e decompor as intenções e razões organizacionais estratégicas como um todo.

Basicamente este modelo é composto de nós e ligações. Os nós no modelo SR têm como base os tipos de Dependum definidos no modelo de dependências estratégicas**: objetivo, tarefa, recurso e objetivo-soft**. Tem-se basicamente dois tipos de classes de ligações: ligação meio-fim e ligações de decomposição de tarefas. Os mesmos são descritos a seguir:

- **Ligação meio-fim (means-ends):** Este tipo de ligação está associada à obtenção de um determinado fim o qual pode ser um objetivo, recurso, objetivo-soft ou tarefa através de um caminho. Este caminho ou meio para obter o fim é geralmente definido em termos de tarefas que são necessárias para atingir/obter o fim desejado.
- **Ligação de Decomposição de Tarefas (task decomposition):** Um **nó tarefa** é ligado aos seus nós **componentes** através de uma ligação de decomposição. Os quatro tipos de nós existentes podem estar ligados,

[3] Todos aqueles que de alguma forma interagem com o sistema sendo desenvolvido.

decompondo um nó tarefa através deste tipo de ligação. Permite-se com este tipo de ligação a decomposição em unidades menores de um nó tarefa para representar de forma mais detalhada as razões associadas com a realização da tarefa. Estas decomposições por sua vez podem estar ligadas através de ligações de **dependência** a outros nós pertencentes à decomposição de outros atores. Isto é normalmente necessário sempre que o relacionamento for relevante para a compreensão das razões e intenções organizacionais.

Neste modelo, uma rotina representa um subgrafo incluindo todas razões bem como os meios para se atingir um fim do ponto de vista de um ator. Através deste modelo pode-se modelar várias alternativas (meios) para se obter fins estratégicos para um ator.

Apresentamos na figura 2 um exemplo de modelo de razões estratégicas para o problema de agendamento de reuniões em uma organização.

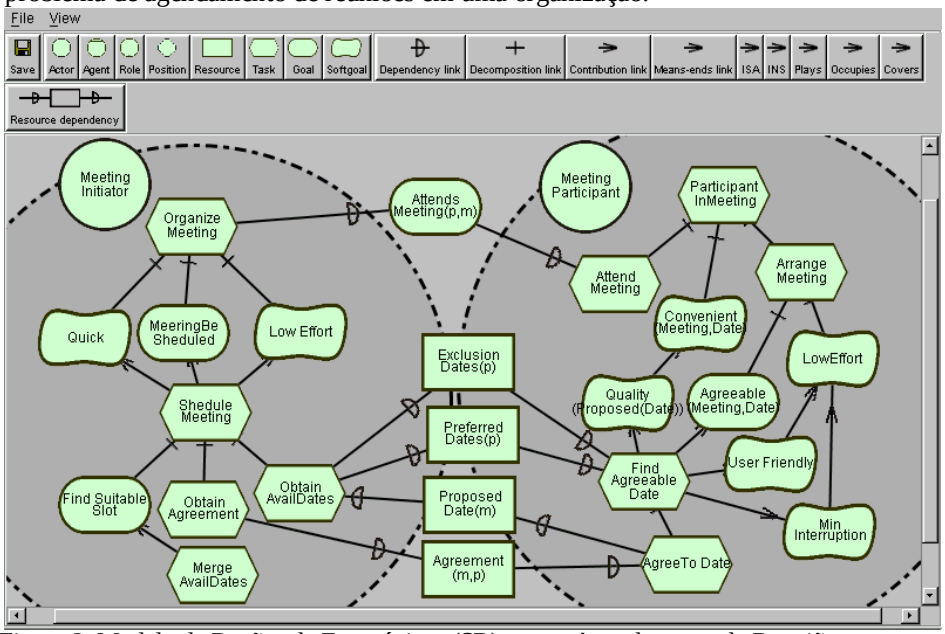

Figura 2. Modelo de Razões de Estratégicas (SR) para o Agendamento de Reuniões, sem considerar a existência de um sistema computacional para realizar o agendamento.

Este exemplo não considera a existência de um sistema baseado em computador para auxiliar no agendamento de reuniões. Por isso, basicamente, uma compreensão inicial do problema inclui a necessidade de que o ator Meeting Initiator deve lidar com um pedido para organizar uma reunião. Desta análise advém a tarefa **Organize Meeting**. Esta característica é importante antes de decidir, por exemplo quais aspectos do agendamento de reuniões poderiam ser cobertos por um possível sistema baseado em computador, quais os tipos de reuniões a serem agendadas, etc. Assim, permite-se explorar as alternativas do processo de agendamento.

Podemos também verificar nesta figura que para realizar uma reunião, o ator Meeting Initiator depende do ator Meeting Participant para que o objetivo comparecer *(AttendsMeeting)* à reunião seja satisfeito. O fato da reunião ser agendada é indicada

pelo subobjetivo *MeetingBeSheduled* da tarefa *Organize Meeting*. Neste ponto, outros subobjetivos poderiam ser incluídos tais como equipamentos a serem reservados para a reunião e memorandos a serem enviados visando relembrar a reunião. Observamos também que a organização da reunião deve ser feita de forma rápida e com menor esforço possível. Estes desejos são representados pelos objetivos-soft *Quick* e Low *Effort*. Para agendar uma reunião, o ator Meeting Initiator obtém informações relativas à disponibilidade de participantes, incluindo datas excluídas e datas preferidas para o agendamento da reunião. O Meeting Initiator (Agendador) então encontra uma data que satisfaça as informações de disponibilidade e obtém concordância dos participantes para confirmar sua participação na reunião. Do ponto de vista do ator Meeting Participant (participante), pode-se se observar que espera-se dos participantes que os mesmos participem ativamente no processo de agendamento (ArrangeMeeting) e após isto compareçam (AttendMeeting) ao encontro. Requer-se que os participantes forneçam as informações de disponibilidade e tentem concordar com datas propostas. Participantes também querem selecionar datas convenientes às suas atividades bem como desejam que o processo de agendamento seja o mais breve e com menor esforço possível, minimizando o número de interrupções do seu trabalho na organização. Estes desejos são representados pelos objetivos-soft Convenient(MeetingDate), LowEffort e MinInterruption, respectivamente.

Após a descrição inicial dos requisitos organizacionais, o engenheiro de requisitos poderá usar outras técnicas, tais com cenários, para complementar a descrição dos requisitos do sistema.

3. Técnicas Baseadas em Cenários

Técnicas baseadas em Cenários têm sido utilizadas na engenharia de software para entender, modelar e validar os requisitos de usuários. Algumas abordagens propondo a utilização de cenários para elicitação e validação de requisitos incluem (Haumer et al. 1998) (Rolland et al. 1998) (Leite et al. 1997) (Jacobson 1995). Na seção 3.1 apresentamos os principais conceitos associados com o uso destas técnicas na engenharia de requisitos e na seção 3.2 descrevemos umas destas técnicas denominada de **Use Cases.** Diagramas de Use Cases são usados como base para a elaboração da proposta deste artigo.

3.1. Utilização de Cenários na Engenharia de Requisitos

Objetiva-se com o uso de cenários descrever as ações em um ambiente relacionadas a um sistema atual ou a um sistema a ser desenvolvido.

Na engenharia de requisitos, dada a dificuldade de elicitar, analisar, negociar e validar os requisitos de software, técnicas baseadas em cenários têm sido consideradas de grande utilidade. Isto porque cenários são construídos do ponto de vista de clientes/usuários, através de uma linguagem facilmente entendida pelos mesmos. As descrições do sistema realizadas através de cenários devem ser entendidas também pelos demais Stakeholders. Assim, o trabalho para elicitar, analisar e validar os requisitos de um sistema pode integrar todos os Stakeholders, num esforço em conjunto, que leva ao desenvolvimento de um documento de requisitos mais completo, consistente e não ambíguo.

No entanto, várias dificuldades surgem na utilização de cenários. Uma destas dificuldades reside no fato que para descrever cenários utilizamos geralmente linguagem natural, a qual pode levar a ambigüidades se o processo de construção não for adequado. Vários trabalhos têm sido realizados apontando problemas na construção de cenários (Ralyté 1999) (Hadad et al. 1999). Lidar com uma grande quantidade de informação e direcionar os esforços na descrição de cenários ainda é um desafio. Além disso, integrar estes cenários de forma que todos os requisitos possam ser definidos e relacionados também é uma tarefa que exige, na maioria dos casos, um alto grau de experiência por parte de engenheiros de requisitos.

Algumas das propostas atuais existentes para o desenvolvimento de cenários incluem: Use Cases (Jacobson 1995) (Schneider et al. 1998), a proposta apresentada pelo projeto CREWS (Ralyté 1999) (Rolland et al. 1998) e a proposta apresentada em Leite et al. (1997). A seguir descrevemos brevemente a técnica de Use Cases, sendo a mesma objeto de estudo de caso neste artigo.

3.2. Use Cases

Use Cases em UML (Booch et al. 1999) são utilizados para descrever o uso de um sistema por atores (Schneider et al. 1998). Um **ator** representa qualquer elemento externo que interage com o sistema. Um **use case** descreve uma seqüência de passos/operações que um usuário realiza quando interage com um sistema visando realizar uma determinada tarefa/objetivo. Assim, o aspecto comportamental de um sistema a ser desenvolvido pode ser descrito. No entanto, a descrição de Use Cases não trata a questão de como esta interação será implementada. Fases posteriores à etapa de engenharia de requisitos tais como Projeto e Implementação focalizarão este aspecto.

Um use case pode gerar vários cenários. Cenários estão para use cases, assim como instâncias estão para classes, significando que um cenário é basicamente uma instância de um use case. Um use case envolve um caso de utilização do sistema por um ator. Neste caso de utilização/uso, vários caminhos podem ser seguidos dependendo do contexto na execução do sistema. Estes caminhos são os possíveis cenários do use case. Considera-se que o caminho básico para realizar um use case, sem problemas e sem erros em nenhum dos passos da seqüência, é denominado de **cenário primário**. Neste tipo de cenário, a execução dos passos para realizar a funcionalidade básica do use case, é obtida com sucesso. Por outro lado, caminhos alternativos bem como situações de erro, podem ser representados através de **cenários secundários**. Cenários secundários descrevem seqüências alternativas e de erros que podem ocorrer em um cenário primário associado com um caso de uso. Cenários secundários podem ser descritos separadamente ou como extensão da descrição de um cenário primário. Se um cenário secundário é bastante complexo e inclui um conjunto bastante grande de passos, é conveniente descrevê-lo separadamente. Cabe ressaltar que a visão de cenários pode variar de acordo com a abordagem utilizada. Por exemplo, em Leite et al. (1997), o escopo atribuído a cenários é mais amplo. Nessa abordagem, cenários são descrições de situações em um ambiente, as quais evoluem. Parte-se do princípio que cenários evoluem juntamente com o processo de desenvolvimento de software, partindo inicialmente da descrição do macrosistema. Além disso, cenários são naturalmente ligados a um LEL (Léxico Estendido da

linguagem) e uma BMV (Visão de modelo Básico) do baseline de requisitos (Leite et al. 1997).

Outras técnicas também podem ser usadas em UML para refinar fluxos de eventos em Use Cases. A idéia consiste basicamente em incluir relacionamentos que permitam descrever diversos aspectos de comportamento entre use cases. Os relacionamentos apontados em UML incluem:

- Relacionamento do tipo **<<include>>:** Quando for detectado no sistema um conjunto de **passos comuns** a vários use cases, pode-se criar um use case com estes passos, com potencial para ser reutilizado por outros use cases. A idéia consiste em abstrair em um **use case específico**, um comportamento comum a vários use cases, estabelecendo que os demais use cases do sistema podem fazer uso do mesmo (i.e. incluí-lo) quando necessário.
- Relacionamento do tipo **(<<extend>>**): Utilizamos este tipo de relacionamento quando existe uma **seqüência opcional ou condicional** de passos que queremos incluir em um use case. Esta seqüência de passos deve ser descrita em um use case específico que poderá ser utilizado por outros use cases em certo ponto de sua execução. O uso do use case extendido ocorre devido a uma situação de comportamento opcional ou condicional.
- Relacionamento do tipo **<<generalization>>:** generalização entre use cases tem o mesmo significado de generalização entre classes na orientação a objetos. Isto significa que um use case "filho" **herda** o comportamento e estrutura do use case "pai". Considera-se que um use case filho é uma **especialização** do use case pai, podendo adicionar nova estrutura e comportamento bem como modificar o comportamento do use case pai.

Da mesma forma que permite-se o uso do mecanismo de generalização entre use cases, pode-se usar o relacionamento de generalização entre **atores**. O ator 1, por exemplo, pode ser uma especialização do ator 2. Neste caso, o ator 1 herda toda a estrutura e comportamento do ator 2 e pode adicionar nova estrutura e comportamento em relação ao ator 2. Outras técnicas adicionais propostas em UML para representar e descrever melhor use case podem ser vistas em Booch et al. (1999). A figura 3 apresenta as notações básicas utilizadas para descrever Use Cases.

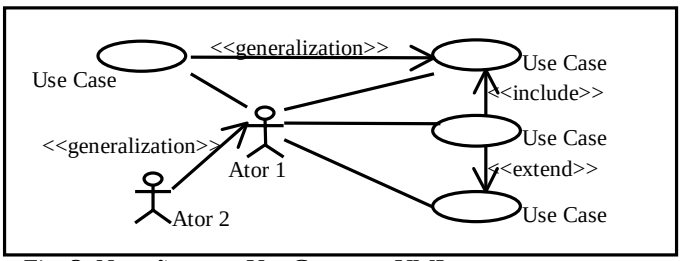

Fig. 3. Notações para Use Cases em UML .

O processo de construção de Use Cases inicia com a descoberta dos atores do sistema e prossegue com a descoberta do(s) caso(s) de uso associados com estes atores. Para cada ator, são encontrados todos os Use Cases relacionados ao mesmo.

Isso ocorre porque cada ator requer do sistema algumas funcionalidades e os passos necessários para obter estas funcionalidades são descritos em Use Cases.

O segundo passo consiste em definir os caminhos básicos (cenários primários) e posteriormente os caminhos alternativos (cenários secundários) para cada um dos Use Cases. O terceiro passo envolve revisar descrições de aspectos comportamentais de use cases encontrando relacionamentos do tipo <<include>>, <<extend>> e <<generalization>>. Esta processo é geralmente realizado adotando-se o princípio de desenvolvimento de software iterativo e incremental (Jacobson et al. 1999). Após definidos todos os Use Cases e atores do sistema, desenvolve-se um modelo de Use Cases utilizando as notações apresentadas na figura 3.

Uma série de heurísticas para auxiliar o engenheiro de requisitos no desenvolvimento de Use Cases, podem ser encontradas em (Schneider et al. 1998) (Booch et al. 1999). Contudo, um dos maiores desafios é o desenvolvimento de Use Cases a partir dos Requisitos Organizacionais previamente modelados.

4. Desenvolvimento de Use Cases a partir dos modelos propostos em i*

Tendo como base as descrições realizadas nas seções anteriores sobre as técnicas i* e Use Cases, podemos apontar algumas diretrizes que auxiliam o trabalho de engenheiros de requisitos no desenvolvimento de Use Cases a partir dos modelos propostos no framework i*. As seções 4.1 e 4.2 descrevem respectivamente, as diretrizes para descoberta de atores bem como os Use Cases com seus cenários primário e secundários em um diagrama de Use Cases.

4.1. Diretrizes para Descoberta de Atores

Como visto anteriormente, um elemento fundamental em ambas as técnicas i* e Use Cases é o **ator.** O processo de construção de Diagramas de Use Cases inicia com a descoberta dos atores do sistema a ser desenvolvido. Este processo de descoberta pode ser auxiliado e facilitado efetuando-se a análise a partir de modelos organizacionais desenvolvidos através da técnica i*. Para tanto, é necessário observar as características associadas com atores nessas duas técnicas. A tabela 1 apresenta de forma resumida alguns conceitos relacionados a atores em ambas as técnicas.

Características de um Ator nas Técnicas i* e Use Cases	
Ator em i*	***Ator em Use Case***
Um ator é uma entidade que realiza ações para obter objetivos no contexto de um ambiente organizacional. Modela-se as intencionalidades no ambiente organizacional como dependências entre atores.	Um ator pode incluir pessoas, sistemas ou máquinas que interagem com o sistema a ser desenvolvido;
Pode-se modelar um ator "genérico" ou "social" em subunidades denominadas de: agente, papel, posição.	Um ator desempenha um papel específico em relação ao sistema.
Atores podem representar partes/componentes do sistema ou o próprio sistema computacional.	Atores são sempre partes externas ao sistema. Eles nunca são partes componentes (internas) do sistema.
Objetiva-se em i* modelar as motivações, intenções e razões que levam atores a realizar ações.	Objetiva-se em Use Cases descrever as ações de um ator interagindo com o sistema.

Tabela 1. Características associadas com um Ator nas técnicas i* e Use Cases.

Tendo em vista as diferenças existentes na conceitualização de ator nas técnicas estudadas (ver tabela 1), é necessário estabelecer algumas diretrizes que conduzam o processo de mapeamento/relacionamento de atores em i* para atores em Use Cases. É importante ressaltar que como visto anteriormente, um ator em i* pode assumir diferentes características inerentes às subunidades**: agente, papel e posição** e estas peculiaridades devem ser observadas no mapeamento de atores entre as técnicas. Algumas diretrizes para observar estas peculiaridades já foram desenvolvidas mas por questões de espaço, neste artigo, apresentaremos somente diretrizes para relacionar atores genéricos/sociais em i* com atores em Use Cases.

A seguir apresentamos algumas diretrizes que permitem apontar candidatos a atores em Use Cases a partir dos atores genéricos definidos em i*.

1. Todo ator genérico, no modelo em i*, deve ser analisado em um possível mapeamento, como um ator em Use Cases;
   Para um melhor entendimento das diretrizes, analisemos, por exemplo, o ator Meeting Participant apresentado no Modelo SD da figura 1;
2. Inicialmente, deve-se verificar se o ator escolhido é **externo** ao sistema. Atores em use Cases nunca são partes do sistema. É necessário observar este aspecto pois na modelagem organizacional pode-se incluir atores representando partes do sistema ou até mesmo uma abstração do software como um todo, o que na técnica de Use Cases não caracteriza um ator;
   Observando a figura 1, verificamos que o ator Meeting Participant participa de um processo de agendamento que não considera a existência de um sistema computacional para o agendamento de reuniões. No entanto, a análise deste ator deve ser realizada observando-se o objetivo de desenvolver um sistema computacional para solucionar esse problema. Deste ponto de vista, o ator Meeting Participant, pode ser considerado externo ao sistema, já que o mesmo interagirá com o sistema a ser desenvolvido fornecendo informações importantes para o sistema. Sob este prisma, o ator em questão é um candidato potencial à ator em Use Cases.
3. Se o ator sendo avaliado é considerado externo ao sistema, conforme passo anterior, é necessário realizar algumas análises do mesmo visando garantir o seu mapeamento para ator no Diagrama de Use cases. Estas investigações incluem:
   3.1. Certificar-se que os relacionamentos de dependência associados com o ator são objetivos/recursos/tarefas/objetivo-softs relevantes que a princípio devem ser tratados pelo sistema a ser desenvolvido de forma indireta ou direta. Isto deve ser observado, tendo em vista a visão bastante ampla da modelagem organizacional o que pode implicar na existência de atores não relevantes no contexto de uso do sistema.

   Observando novamente o ator genérico Meeting Participant (figura 1), verificamos que as dependências associadas com o mesmo, caracterizam-no como importante em um contexto de interação com o sistema. As dependências entre o mesmo e o ator Meeting Initiator, mostram que Meeting Participant tem a responsabilidade de atender e fornecer informações que serão tratadas em um sistema computacional para tratar o problema de agendamento de reuniões. Portanto, o mesmo

possui características que o torna candidato a ator no Diagrama de Use Cases.

3.2. Se o ator assumir apenas o papel de Dependee no modelo organizacional, verificar se as dependências em relação ao mesmo não estão relacionadas à dependências de atores do sistema em relação ao próprio software. Neste caso, o ator talvez seja o software ou parte componente do software a ser desenvolvido, não enquadrando-se portanto como ator em Use Case.

Podemos observar na figura 1, que o ator Meeting Participant assume tanto o papel de Depender como de Dependee no modelo proposto e portanto não se enquadra nesta análise. Desta forma, o ator Meeting Participant pode ser anotado como ator no Diagrama de Use Cases.

3.3. Em relacionamentos do tipo **ISA** entre atores genéricos tais como: o **ator1** é (ISA) do tipo **ator2,** ambos os atores devem ser avaliados conforme diretrizes anteriores. Normalmente ambos os atores são considerados atores em um Diagrama de Use Cases. No entanto, deve-se observar que este tipo de relacionamento é modelado no Diagrama de Use Cases através do mecanismo de **generalização,** enquadrando-se o ator 2 como uma especialização do ator 1. A notação no diagrama de use cases para este tipo de relacionamento entre atores foi apresentada na seção 3.2.

Por exemplo, na figura 1, o relacionamento ISA entre Important Participant e Meeting Participant indica que ambos atores são candidatos a atores em use Cases. Conforme diretrizes anteriores já aplicadas na avaliação do ator Meeting Participant, o mesmo pode ser considerado como ator em Use Cases. Da mesma forma, aplicando-se as mesmas diretrizes ao ator Important Participant, pode-se apontar o mesmo como ator no Diagrama de Use Cases. O relacionamento entre ambos deve ser do tipo **<<generalization>>**. O ator Important Participant deve ser anotado como uma especialização de Meeting Participant.

De forma geral, as diretrizes acima permitem a elaboração de uma lista de candidatos a atores no desenvolvimento de use cases. Outras heurísticas existentes em (Schneider et al. 1998) devem ser usadas para auxiliar neste processo, bem como o própria experiência de engenheiros de requisitos/software. Após este processo a lista de atores deve ser estabilizada. O que é importante ressaltar é que nesta abordagem o engenheiro de requisitos pode iniciar o desenvolvimento de use cases já familiarizado com intenções e objetivos organizacionais estratégicos. O fato de se desenvolver um modelo organizacional no processo de engenharia de requisitos possibilita esta visão bastante útil no desenvolvimento de use cases.

4.2. Diretrizes para Descoberta de Use Cases e seus Cenários

Continuando o processo de desenvolvimento de Use Cases a partir de modelagem organizacional, é necessário observar tanto o modelo de Dependências Estratégicas quanto o modelo de Razões Estratégicas para encontrar possíveis candidatos a use cases (casos de uso) para cada ator. Uma análise mais detalhada

destes modelos permite-nos visualizar a possibilidade de a partir das dependências e razões estratégicas associadas com atores em i* apontar os Use Cases para estes atores bem como também o cenário primário e os cenários secundários de cada use case.

O modelo de Dependências Estratégicas (SD) representa basicamente dependências entre atores do tipo recurso, objetivo, tarefa e objetivo-soft e se observarmos melhor estes relacionamentos, verificamos que de forma indireta ou direta os mesmos estão relacionados à atividades e funções realizadas no ambiente organizacional. Se o nosso objetivo é desenvolver uma solução computacional para todas ou uma parte destas funções, é razoável afirmar que muitos destes relacionamentos (dependências e ligações) podem levar a use cases ou passos (descrição) de um use case. Além disso, encontramos também no modelo de dependências estratégicas, dependências do tipo objetivo-soft que na engenharia de requisitos podem ser denominadas de requisitos não funcionais, os quais por sua vez, estão na maioria dos casos direta ou indiretamente associados a Use Cases. Portanto, além dos Use Cases (funcionalidade) visualiza-se também a possibilidade de que aspectos não funcionais, os quais podem ser anexados (através de uma descrição textual resumida) a Use Cases, também possam ser obtidos a partir de i*.

Nessa mesma análise, o modelo de Razões Estratégicas (SR), o qual descreve os aspectos internos relevantes a processos e intenções de atores, é fonte de informações para o desenvolvimento de Use Cases. Cada ator pode ter associado decomposições e ligações do tipo meio-fim e decomposição de tarefas (ver seção 2.2) as quais visam descrever de forma mais detalhada passos para obtenção de objetivos, recursos, tarefas e objetivos-soft de um ator. Com base nestas ligações podemos extrair outros use cases não descobertos a partir do modelo de dependências estratégicas bem como também passos internos a use cases, os quais representam tipicamente cenários secundários e o cenário principal em um use case.

Esta tarefa de observar os modelos em i* e extrair informações para Use Cases não é trivial pois ambas as técnicas permitem enfocar requisitos de software sob diferentes perspectivas. O modelo de Use Cases é um modelo criado com o fim de representar Use Cases (Casos de Uso) do sistema e consequentemente os requisitos do mesmo. Já a modelagem organizacional, implicitamente permite que informações consideradas da organização possam ser utilizadas tendo um enfoque voltado para a descrição do sistema. No entanto, como os modelos organizacionais podem representar alternativas para a obtenção de objetivos organizacionais, uma das vantagens deste mapeamento é que o engenheiro de requisitos além de estar envolvido conscientemente com os objetivos organizacionais pode também optar, juntamente com o cliente, por opções a serem descritas em use cases que obtenham da melhor maneira possível as metas e objetivos da organização e/ou do cliente. Assim, aspectos antes não considerados na descoberta de Use Cases podem ser avaliados, analisados e incluídos na descrição dos mesmos.

A seguir apresentamos algumas diretrizes que podem auxiliar no trabalho de descoberta dos **use cases para cada ator** (descoberto utilizando as diretrizes apresentadas anteriormente) a partir da modelagem organizacional. Ambos os modelos de i* devem ser observados.

1. Inicialmente devemos analisar para cada **ator** todas as dependências que são atribuídas ao mesmo como **Dependee**. O ator é observado do ponto de vista de quais dependências (Dependum) ele é responsável e deve atender

para o(s) Depender(s). Cada Dependum é candidato a Use Case do ator. As diretrizes abaixo auxiliam nesta análise:

1.1. Se o Dependum for do tipo **Objetivo**, em grande parte dos casos este Objetivo pode ser considerado um use case do ator. No entanto, para uma maior garantia, deve-se analisar a dimensão desse objetivo pois em algumas situações o mesmo pode ser parte integrante de outro Use Case, possivelmente coberto por outro objetivo mais amplo. A nomenclatura do Dependum (objetivo) deve ser adaptada para representar melhor o conceito de use case.

Analisemos, por exemplo, as dependências do ator **Meeting Participant** na figura 1. Observando esta figura, verificamos a existência da dependência do tipo Objetivo AttendsMeeting(p,m) que este ator deve satisfazer para o ator Meeting Initiator. Avaliando AttendsMeeting(p.m), temos que o mesmo pode ser considerado um use case do Ator Meeting Participant, pois é um objetivo bastante amplo que implica em vários passos que devem ser realizados pelo Meeting Participant para poder atender à reunião. Portanto, podemos apontar um use case **AttendsMeeting** que conterá os passos envolvidos pelo participante para atender à reunião.

1.2. Se o Dependum for do tipo **Recurso**, geralmente esse Dependum ou mais comumente a junção de algumas dependências desse tipo, resulta em um Use Case para o ator. Cabe aqui observar se os recursos (dependums desse tipo para o ator) estão associados ou são necessários a uma funcionalidade (caso de uso) específica do sistema. Neste caso, um nome adequado para o Use Case deve ser anotado representando melhor a idéia de junção dos recursos congregados no Use Case.

Analisando novamente a figura 1, do ponto de vista do Ator **Meeting Participant** como Dependee, podemos observar que existem duas dependências do tipo recurso ExclusionDates(p) e PreferredDates(p). Estas dependências podem ser consideradas para compor um Use Case denominado de **EnterAvailDates**. Esta análise é feita com base na observação de que ambas as dependências representam datas que estabelecem as datas disponíveis de um participante para agendar uma reunião. Assim, consideramos a existência de um Use Case de MeetingParticipant que contenha os passos para definir datas possíveis de Agendamento de uma reunião.

1.3. Se o Dependum for do tipo **tarefa** o mesmo princípio adotado para os outros dois tipos de dependência deve ser adotado. Uma tarefa que claramente inclua outros passos para levá-la a cabo e necessite ser decomposta devido a sua complexidade, pode ser considerada um Use Case. No entanto, se a tarefa for bastante simples geralmente poderá ser agregada a outras tarefas com um fim em comum, sendo então este conjunto de tarefas associadas com o ator, considerado um Use Case. A nomenclatura também deve ser adaptada para representar um caso de uso pelo ator envolvido.

No exemplo da figura 2, a tarefa OrganizeMeeting do ator Meeting Initiator pode ser candidata a Use Case, já que a mesma é uma abstração em alto nível do processo de organização de uma reunião. Um Use Case denominado de **OrganizeMeeting** poderia ser definido para o ator **MeetingInitiator**, representando os passos de uso do sistema tendo em vista a necessidade de organizar uma reunião.

1.4. O Dependum do tipo **Objetivo-soft** é associado na modelagem organizacional a uma meta que ainda não está totalmente definida (por exemplo que a reunião seja rápida e produtiva). Portanto, um objetivo-soft corresponde a um requisito não funcional que deve-se ser associado aos Use Cases descobertos a partir dos outros tipos de Dependum.

Por exemplo, na figura 1, o objetivo-soft **Assured(AttendsMeeting(ip,m)** entre Meeting Initiator e Important Participant estabelece que há um desejo de garantir o comparecimento do participante importante à reunião. Dessa forma, este objetivo-soft pode ser satisfeito ou não dependendo da garantia dada pelo ator Important (será que a palavra dele é suficiente ou se faz necessário uma confirmação formal dele). Assumindo a existência de um Use Case denominado de AttendsMeeting para o ator Important Participant, poderíamos associar este requisito não funcional ao Use Case AttendsMeeting associado com o ator (por exemplo, indicação que este compromisso é importante e não pode ser cancelado). Uma descrição deste requisito poderia ser anexada ao Use Case.

2. Anotados os candidatos iniciais a Use Cases para cada ator, conforme passo anterior, deve-se realizar uma análise mais detalhada observando-se o modelo de Razões Estratégicas (SR). Este modelo associa com cada ator, decomposições que envolvem as razões estratégicas do mesmo em relação às dependências com outros atores. Pode-se então a partir desta decomposição extrair informações referentes a descoberta de outros Use Cases e principalmente informações sobre os passos (tarefas) internos à realização de Use Cases para o ator. Ligações de decomposição de tarefas e ligações do tipo meio-fim podem representar estes passos. Deste modelo também podem ser coletadas informações relacionadas à requisitos não funcionais (objetivo-softs) desejados para os Use Cases do ator.

   Por exemplo, observando-se o modelo de Razões Estratégicas da figura 2, do ponto de vista do ator **Meeting Initiator**, verificamos que a tarefa Shedule Meeting foi decomposta em ObtainAvailDates, FindSuitableSlot e ObtainAgreement. Tendo em vista o desenvolvimento de um sistema computacional para realizar agendamentos de reuniões, poderíamos adotar que estes são os passos de alto nível necessários para realizar um agendamento de uma reunião (Shedule Meeting). Desta forma, um Use Case denominado de **SheduleMeeting** poderia ser definido para o ator Meeting Initiator. Este Use Case poderia conter os passos (descrição do cenário principal) referentes à necessidade de obter as datas disponíveis dos participantes para uma reunião (ObtainAvailDates); encontrar com base nesta

disponibilidade dos participantes, as datas em que a reunião poderia ser agendada (FindSuitableSlot); e obter a concordância dos participantes em relação a uma data proposta para a reunião (ObtainAgreement). Dessa forma, a partir do modelo SR poderíamos abstrair Use Cases bem como descrições de cenários dos mesmos.

Estas diretrizes necessitam ser melhoradas e estendidas mas são suficientes para apresentar a viabilidade do desenvolvimento de uses cases a partir de modelagem organizacional. A seguir apresentamos um exemplo da aplicação destas diretrizes possibilitando uma compreensão melhor da nossa proposta.

4.3. Aplicando as diretrizes a um estudo de caso.

Assumamos que o nosso objetivo seja encontrar os Use Cases para um sistema computacional de **agendamento de reuniões** em uma organização. Adotemos também que a notação utilizada para o desenvolvimento de Use Cases é a proposta em UML (Booch et al. 1999).

A figura 6 descreve um exemplo de modelo de dependências estratégicas para um agendamento de reuniões, o qual inclui um sistema baseado em computador denominado de **Meeting Sheduler** (Agendador de Reuniões).

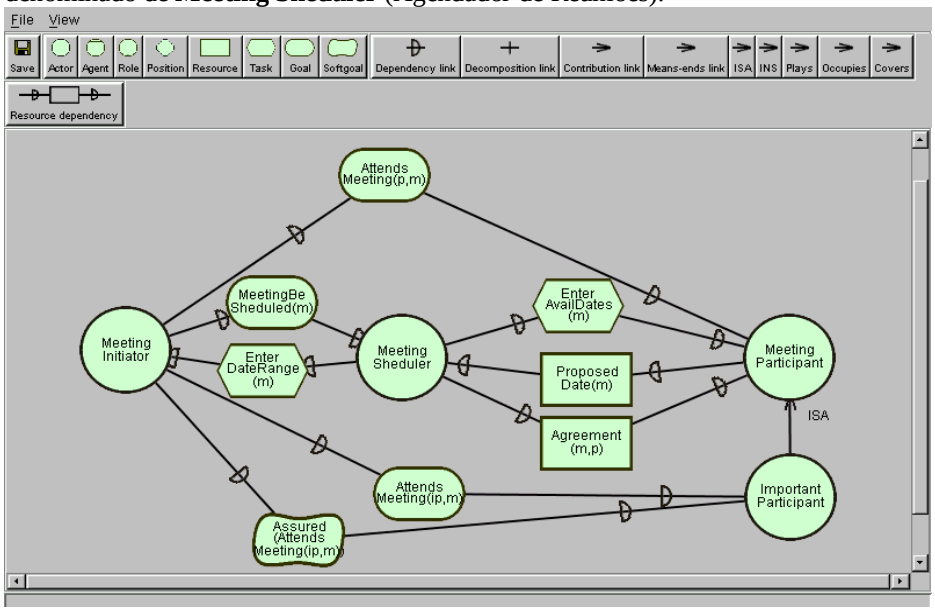

Figura 6. Modelo de dependências Estratégicas para Agendamento de Reuniões, considerando a existência de um sistema computacional Meeting Sheduler.

Lembremos que na seção 2.1, apresentamos na figura 1, um modelo de dependências estratégicas para agendamento de reuniões, o qual desconsidera a existência de sistema computacional (Meeting Sheduler). Na modelagem organizacional é comum desenvolver várias visões como neste caso, procurando representar melhor as intenções e motivações para um determinado processo. Diante disto, a análise para descoberta de Use Cases deve considerar todos os modelos

organizacionais produzidos. Em nosso estudo de caso consideraremos os dois modelos de Dependências Estratégicas (SD) apresentados na figura 1 e 6.

Outro modelo que será base para o nosso estudo de caso é apresentado na figura 7. Este representa um modelo de Razões Estratégicas (SR) para o modelo apresentado na figura 6. Neste modelo são apresentadas as Razões Estratégicas associadas com os três atores organizacionais para a realização do processo de agendamento de reuniões. Parte do trabalho de Agendamento de Reuniões é atribuído a um sistema computacional denominado de Meeting Sheduler e o mesmo é decomposto e relacionado com elementos componentes dos outros atores.

Com base nos modelos apresentados nas figuras 1, 6 e 7 podemos aplicar as diretrizes para desenvolver os Use Cases conforme passos descritos nas seções 4.1 e 4.2. É importante ressaltar que estes modelos organizacionais são apresentados conforme constam originalmente em Yu (1995).

Observando a figura 6, podemos encontrar candidatos a atores para o desenvolvimento de Use Cases. Seguindo as diretrizes definidas na seção 4.1, verificamos que um dos atores no modelo analisado fere o exposto no passo 2 dessas diretrizes. O ator **Meeting Sheduler** representa um sistema computacional, o qual é o próprio sistema que objetivamos desenvolver. Portanto, o mesmo não pode ser considerado como candidato a ator em Use Cases. Os outros atores são considerados relevantes pois suas dependências estratégicas, em uma primeira análise, referem-se a aspectos relevantes para o desenvolvimento do sistema computacional para agendamento de reuniões. Desta forma a lista de candidatos a atores em Use Cases inclui: **Meeting Initiator, Meeting Participant** e **Important Participant**. Observando melhor estes candidatos verificamos que Important Participant é um tipo (relacionamento ISA) de participante, sendo que as funcionalidades associadas com esse ator são quase todas já cobertas pelo ator Meeting Participant. Conforme diretriz 3.3 apresentada na seção 4.1, vamos considerar este ator uma **especialização** de Meeting Participant e estabelecer este tipo de relacionamento entre os mesmos, utilizando a notação padrão em Diagramas de Use Cases (Booch et al. 1999).

Para efeitos de estudo e entendimento da nossa proposta de descoberta de Use Cases a partir de modelos organizacionais, vamos considerar apenas os Use Case para os atores **Meeting Initiator** e **Meeting Participant**.

Prosseguindo na descoberta de Use Cases a partir dos modelos organizacionais em i*, tem-se que o próximo passo é descobrir e relacionar os Use Cases para cada ator. Assim, seguindo as diretrizes apresentadas podemos constatar:

- Para o ator **Meeting Participant** podemos apontar alguns candidatos a Use Case oriundos dos relacionamentos de dependência do ator como **Dependee**. Inicialmente, analisando a dependência do tipo tarefa *EnterAvailDates(m)* do Modelo SD da figura 6 e as dependências do tipo recurso *ExclusionDates(p)* e *PreferredDates(p)* no Modelo SD figura 1 podemos visualizar que estas dependências consistem basicamente no Use Case **EnterAvailDates,** o qual representa que o ator Meeting Participant necessita estabelecer um conjunto de datas possíveis para Sheduler Meeting (Sistema Agendador de Reuniões) com base em datas preferidas e excluídas por motivos pessoais. Entende-se que esta atividade envolve vários passos e portanto possui características que justificam a sua inclusão como um use case.

Podemos também apontar o Use Case: **AttendsMeeting,** tendo como base de observação a dependência do tipo objetivo *AttendsMeeting(p,m)* (figura 6), o qual representa a necessidade do participante atender a reunião. Isto implica em que assumir que vários passos são necessários para obter este objetivo.

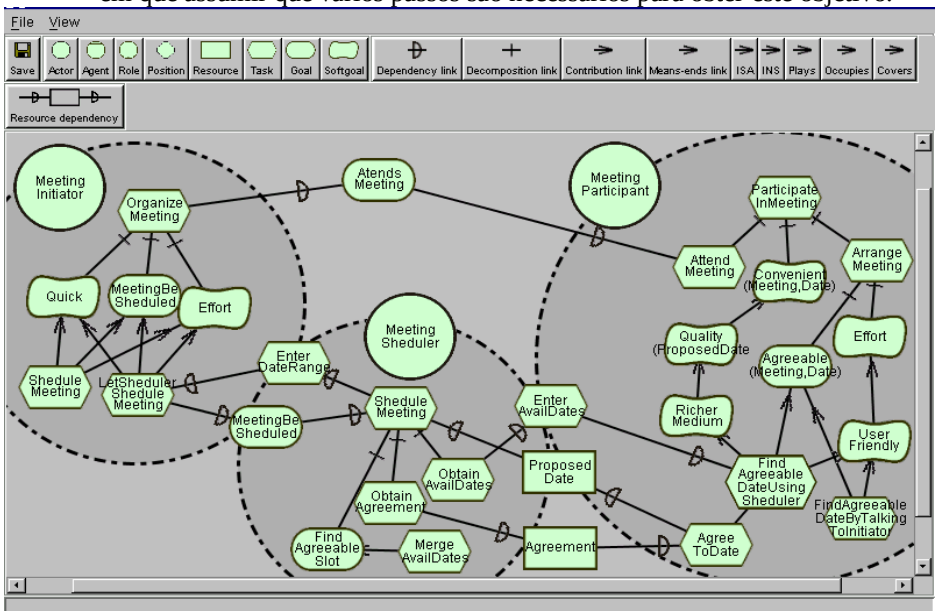

Figura 7. Modelo de Razões Estratégicas para um sistema de Agendamento de Reuniões.

- Para encontrar os candidatos a Use Cases para o ator **MeetingInitiator** deve-se adotar o mesmo princípio do passo anterior. Do ponto de vista do ator como Dependee, podemos apontar o Use Case **EnterDateRange,** tendo como base a observação da dependência *EnterDateRange(m)* (figura 6). O ator Meeting Initiator deve estabelecer um intervalo de datas possíveis para o agendamento da reunião. Desta forma, um Use Case com este fim deve ser considerado, tendo em vista que esta atividade pode incluir vários passos, incluindo aspectos tais como intervalos de datas relacionadas com o tipo de reunião a ser agendada e prioridades de reuniões.

  Como parte do esforço de agendamento de uma reunião foi atribuído a um sistema computacional denominado de Meeting Sheduler, é importante neste momento também observar o modelo da figura 7, que expressa as razões estratégicas envolvidas entre os atores. As razões estratégicas do ator Meeting Initiator, representam as responsabilidades e tarefas associadas com o mesmo no processo de organização de uma reunião. Destes relacionamentos podemos apontar a tarefa OrganizeMeeting como origem para um candidato a Use Case, já que a mesma pode ser considerada uma abstração de alto nível de um processo de organização e agendamento de reuniões. Sob este ponto de vista, um Use Case denominado de **OrganizeMeeting**, poderia conter os passos necessários para organizar uma

reunião e realizar o agendamento. (Estes passos são descritos no cenário primário do Use Case).

Da mesma forma, em um nível mais baixo de abstração, podemos verificar que a dependência **MeetingBeSheduled** (figuras 6 e 7) entre Meeting Initiator e Meeting Sheduler (o qual é um sistema de software) aponta para a definição de um Use Case que trate aspectos de interação e entre o ator e o sistema para a realização do agendamento. Um Use Case denominado de **Shedule Meeting** poderia representar este uso do sistema pelo ator, descrevendo os detalhes do processo de agendamento de reuniões. A troca de informações entre ator Meeting Initiator e o sistema Meeting Sheduler bem como a decomposição da tarefa **Shedule Meeting** (associada com Meeting Sheduler na figura 7) apontam para a necessidade de compreender bem o processo organizacional antes de descrever os uses cases que possam ser derivados destes relacionamentos.

- Em uma análise mais aprofundada dos modelos, detectamos que há necessidade de garantir que os acessos ao sistema tanto do ator Meeting Initiator (Agendador) quanto de Meeting Participant (Participantes) seja garantido e controlado. Mudanças de informações sobre disponibilidade de datas (tarefa *EnterAvailDates* na figura 7*)*, por exemplo, só devem ser realizadas pelo participante sendo consultado. Tendo em vista esta necessidade é conveniente criar um Use Case denominado de **ValidateUser** que possa ser incluído por outros Use Cases. Assim como outros Use Cases que podem ser descobertos no sistema, ValidateUser não têm um elemento (dependência) associado diretamente nos modelos de dependência e razões estratégicas descritos nas figuras 1, 6 e 7. Por este motivo, a análise de descoberta de Use Cases para atores ainda requer um certo grau de experiência do engenheiro de requisitos bem como auxílio de outras heurísticas tais como as apresentadas em Schneider et al. (1998).

Diante das considerações acima realizadas, temos na figura 8 a representação de atores juntamente com seus relacionamentos com os Use Cases descobertos a partir dos modelos organizacionais apresentados.

Os Use Cases da figura 8 ainda podem ser alterados e ou complementados dependendo da visão e da experiência do engenheiro de requisitos. Modelagens desta natureza podem variar visando facilitar o entendimento bem como estabelecer um acordo entre clientes e desenvolvedores em relação aos requisitos do sistema.

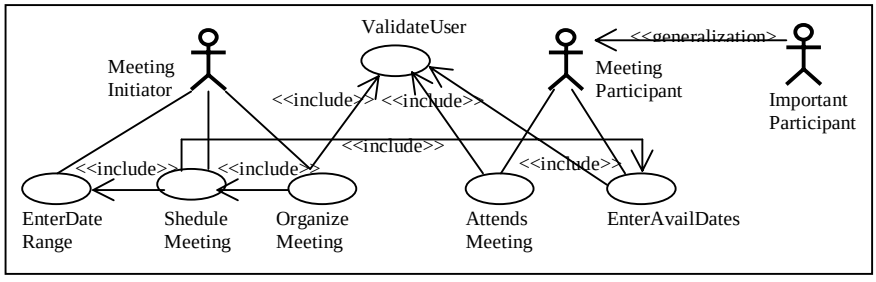

Figura 8. Use Cases para um sistema de agendamento de reuniões.

Tendo em mãos os Use Cases descobertos para o sistema de agendamento torna-se necessário definir o cenários primário e secundários (se necessários) bem como os relacionamentos do tipo <<include>>, <<extend>> e <<generalization>> relacionados com os mesmos. Neste ponto, o modelo de dependências estratégicas e principalmente o modelo de razões estratégicas servem de fonte de informação para a descrição dos cenários e estabelecimento dos relacionamentos.

Por exemplo, o use case SheduleMeeting representa o uso do sistema pelo ator Meeting Initiator para realizar um agendamento de uma reunião. Este use case deveria conter todos os passos necessários ao agendamento da reunião, tendo inicio quando o ator fornece ao sistema informações tais como o intervalo de datas entre as quais deseja-se agendar a reunião. Outras tarefas que o ator espera que o sistema realize para que o agendamento seja efetuado também devem ser descritas. O sistema deve, com base no intervalo de datas fornecido pelo ator Meeting Initiator, encontrar datas disponíveis por todos os participantes para a reunião bem como posteriormente elaborar uma lista de datas de consenso entre as quais será escolhida uma data a ser proposta e acordada. Este processo deve resultar no agendamento de uma data de consenso para a reunião e posteriormente na confirmação desta data por todos os participantes. Desta forma, para o Use Case SheduleMeeting poderíamos ter o cenário primário com os seguintes passos :

Use Case: **SheduleMeeting**

Passos

1. O Use Case inicia com o ator Meeting Initiator fornecendo ao sistema um intervalo de datas entre as quais pode-se agendar a reunião; (O Use Case **EnterDateRange** é incluído <<include>> neste passo).
2. O sistema deve solicitar dos participantes (ator Meeting Participant) uma lista de datas possíveis para a reunião, tendo como informação o intervalo proposto pelo ator Meeting Initiator; (O Use Case **EnterAvailDates** é incluído <<include>> neste passo).
3. O sistema deve encontrar uma lista de consenso filtrando informações com base nas datas possíveis enviadas pelos participantes e no intervalo de datas propostas pelo ator Meeting Initiator;
4. O ator Meeting Initiator com base na lista de consenso e nas prioridades estabelecidas pela organização escolhe uma data de agendamento e fornece esta data ao sistema para a realização da consulta aos participantes;
5. O sistema solicita concordância dos participantes em relação à data agendada para a reunião.

As informações para descrição deste use case tem como fonte principal o **modelo de Razões Estratégicas (SR)** apresentado na **figura 7**. A informação que fundamenta o passo 1 deste Use Case é extraída da dependência do tipo tarefa EnterDateRange, estabelecendo a necessidade de que o ator Meeting Initiator forneça um intervalo de datas entre as quais deve-se realizar o agendamento da reunião. Consideramos que este processo de estabelecer um intervalo de datas de agendamento, é composto de vários passos, os quais estão contidos no use case **EnterDateRange.** Por este motivo, neste passo, o use case EnterDateRange é incluído <<include>>.

Os passos 2 e 3 são extraídos das decomposições da tarefa Shedule Meeting (associada com Meeting Sheduler na figura 7). O passo 2 é extraído da análise da tarefa ObtainAvailDates. O use case **EnterAvailDates** é incluído <<include>> pois representa os passos necessários para a obtenção das datas de agendamento disponíveis pelos participantes. O passo 3 tem origem na observação do objetivo FindAgreeableSlot e da tarefa MergeAvailDates.

O passo 4 advém da observação da dependência do tipo recurso ProposedDate em relação à tarefa Shedule Meeting. Consideramos que a data proposta para agendamento deve ser escolhida pelo ator Meeting Initiator, com base, por exemplo, em prioridades de reuniões da organização. Esta informação não está explicita nos modelos organizacionais apresentados, configurando um exemplo de que modelos organizacionais podem não conter informações importantes para a descrição de Use Cases. O passo 5 resulta da necessidade do sistema obter junto aos participantes da reunião a concordância em relação à data escolhida para agendamento. Esta informação advém da observação da tarefa ObtainAgreement.

Os demais use cases descobertos no nosso estudo de caso, podem ser descritos de forma análoga ao processo e diretrizes adotados na descrição do use case Shedule Meeting.

Desta forma, observando-se os modelo organizacionais e adotando-se as diretrizes propostas neste artigo, podemos extrair informações bastante úteis no desenvolvimento de Use Cases. Como dito anteriormente, a descoberta dos Use Cases bem como a descrição dos mesmos pode ser melhorada a partir de uma observação mais detalhada dos modelos organizacionais bem como do uso de heurísticas propostas na literatura para o desenvolvimento de Use Cases.

5. Considerações Finais e Trabalhos Futuros

Apresentamos neste artigo, algumas diretrizes que permitem o uso de modelos organizacionais em i* no desenvolvimento de cenários sob a forma de Use Cases. Ambas as técnicas i* e Use Cases foram descritas e as diretrizes propostas foram aplicadas parcialmente a um estudo de caso de um sistema de agendamento/escalonamento de reuniões. A partir do estudo de caso foi possível observar que as informações existentes tanto no modelo de dependências estratégicas quanto no modelo de razões estratégicas podem de forma direta ou indireta servir de base para o desenvolvimento de Use Cases.

Além disso, permite-se que o engenheiro de requisitos, a partir de uma observação mais detalhada dos modelos organizacionais, possa optar pela melhor alternativa para desenvolver o software bem como concentrar-se nos Use Cases que de fato representam a satisfação dos objetivos organizacionais. No desenvolvimento tradicional, cenários de uma forma geral não consideram de forma eficiente motivações, intenções e alternativas para o desenvolvimento de sistemas. O uso de modelos organizacionais auxilia neste sentido.

Com este tipo de abordagem integrando modelos organizacionais e Use Cases perguntas tais como: de que forma o sistema pretendido irá satisfazer os objetivos da organização, por que ele é necessário, quais as alternativas existentes e quais as implicações das alternativas para as várias partes interessadas, podem ser melhor respondidas, tratadas e as soluções incorporadas no desenvolvimento de sistemas.

Alguns trabalhos relacionados incluem propostas centradas no desenvolvimento de software orientado a objetivos apresentadas em Mylopoulos et al. (1999).

Em trabalhos futuros pretendemos tratar os seguintes aspectos:

- Estudo e melhoramento das propostas de diretrizes apresentadas neste artigo. Consideramos este trabalho inicial e motivador para o desenvolvimento de um framework mais amplo no qual todos os aspectos relevantes ao desenvolvimento de Use Cases a partir de modelagem organizacional sejam cobertos;
- Extensão das propostas às outras técnicas baseadas em cenários apresentadas em Leite et al. (1997) e no projeto CREWS (Ralyté 1999) (Ralyté et al. 1999) (Rolland et al. 1998);
- Avaliar e validar as propostas de diretrizes através de estudos de caso reais.

6. Referências Bibliográficas

Booch, G., Jacobson, I., Rumbaugh, J., *"The Unified Modeling Language User Guide"* , Addison-Wesley (1999).

Bubenko, J. A., Extending The Scope of Information Moldeling, Proc. 4th Int. Workshop on the Deductive Approach to Information Systems and DataBases, Lloret-Costa Brava, Catalonia, Sept. 20-22, 1993, pp73-98.

Hadad, G. D.S., Doorn, J.H., Kaplan, G.N., Leite, J.C.S.P., Enfoque Middle-Out en la Construcción e Integración de Escenarios, nos anais do Workshop em Engenharia de Requisitos, Buenos Aires, Argentina, 9-10 September, 1999.

Haumer, P., Pohl, K., Weidenhau´pt, K., Requirements Elicitation and Validation with Real World Scenes, IEE Transactions on Software Engineering, Vol 24, No 12, Special Issue on Scenario Management, December, 1998.

Jacobson, I., Booch, G., Rumbaugh, J., *The Unified Software Development Process*, Addison-Wesley (1999).

Jacobson, I., *Object Oriented Software Engineering: A Use Case Driven Approach*, Addison-Wesley (1995).

Leite, J.C.S.P, Rossi, G., Balaguer, F., Maiorana, V., *Enhancing a requirements baseline with scenarios.* In Proceedings of the Third IEEE International Symposium on Requirements Engineering – RE97, pages 44-53. IEEE Computer Society Press, January, 1997.

Mylopoulos, J., Chung, L., and Yu, E., "From Object-Oriented to Goal-Oriented Requirements Analysis," *Communications of the ACM*, **42**(1), pp. 31-37. January 1999.

Ralyté, Jolita., Rolland, C., Plihon, V., *Method Enhancement With Scenario Based Techniques*, To appear in *Proceedings* of CAISE 99, 11th Conference on Advanced Information Systems Enguneering Heidelberg, Germany, June 14-18, 1999, (CREWS Report Series 99-10).

Ralyté, Jolita., *Reusing Scenario Based Approaches in Requirements Enginnering Methods: Crews Method Base*, To appear in *Proceedings* of REP'99, 1st International Workshop on the Requirements Engineering Process, Florence, Italy, September 1999, (CREWS Report Series 99-12).

Rolland, C., Souveyet, C., Achour, C. B., *Guiding Goal Modeling Using Scenarios*, IEE Transactions on Software Engineering, Vol 24, No 12, Special Issue on Scenario Management, December, 1998.

Schneider, G., Winters, J. P., *Applying Use Cases: a practical guide*, Addison Wesley, (1998).

Yu, Eric, *Modelling Strategic Relationships for Process Reengineering*, Phd Thesis, University of Toronto, (1995).